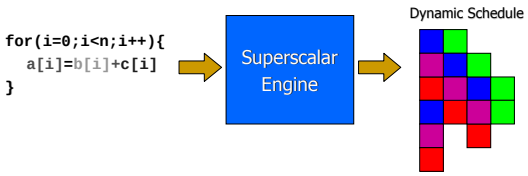
Finding Parallelism for Future EPIC Machines

Matthew Iyer†, Chinmay Ashok†,
Neil Vachharajani‡, Joshua Stone †*,
Daniel Connors†, Manish Vachharajani†

†University of Colorado, Boulder, CO
‡Princeton University, Princeton, NJ
*Now at Intel Corporation, Santa Clara, CA

Parallelism in Current Architectures

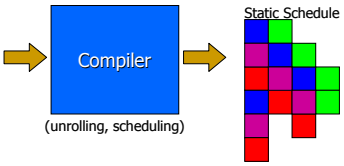
```
for(i=0;i<n;i++){
  a[i]=b[i]+c[i]
}
```



- Traditional models focus on exploiting near ILP in hardware
  - Near ILP: within range of 2K of instructions
  - Speculation and multithreading [Lam '92, Austin '92, Wall '93]
  - Reduced memory latency [Jouppi '97]
  - Register renaming [Wall '93, Postiff '98, Lee '00]
  - Large instruction windows [Austin '92]
- Kilo-instruction machines [Valero '04] proposed

Parallelism in Current EPIC Architectures

```
for(i=0;i<n;i++){
  a[i]=b[i]+c[i]
}
```



- EPIC models focus on exploiting near-ILP using software
  - Larger compiler optimization and analysis scope
    - Inlining, loop-unrolling, etc.
  - Less able to exploit dynamic properties of execution
    - e.g., variable memory latency [Sias ISCA '04]
    - Difficult to tolerate cache miss penalties

Amount of Parallelism

- Several limit studies
  - Flynn's bottleneck of instructional level parallelism [Flynn '70]
  - **Control flow** limits parallelism [Lam '92]
  - Comprehensive study [Wall '93] shows how various factors affect **ILP at the microarchitectural level**
  - ILP limits in response to increasing **memory latency** and **branch misprediction penalty** [Jouppi '97]
  - Stack aliasing : **unnecessary true data-dependences** in stack pointer manipulation [Postiff '98, Lam '92]
  - **Limitations of procedure and loop boundaries** on ILP of an EPIC architecture [Lee '00]

Characterizing Parallelism

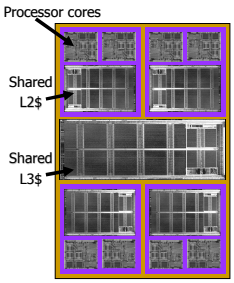
- Ever-changing hardware and software conditions
  - Workloads and programming concepts
  - Compiler technology (analysis and optimization)
- Prior studies focus on measurement and lack deep analysis to guide research
- Ideal parallelism: upper bound on the inherent parallelism in the implementation of an algorithm
  - Programming model, compiler, and ISA are involved
  - Examine complete program execution
- Studies must be **comparative** to allow conclusions that **characterize the sources of ILP**

Distant Parallelism

- Eliminate barriers to observing parallelism
  - Stack activity
  - Control flow
  - Resources
- Study different effects of ISAs and compilers
- Discovery: critical dependences release many distant parallel instructions
  - Independent execution regions exist and form local dependence chains

```
Cfg_t * cfg;
Func_t * func;
Program_t * prog;
...
prog = read(binary);
...
while(func=prog->first;
      func!=prog->last;
      func=func->next) {
  cfg=init(func);
  dom_analysis(cfg);
  pdom_analysis(cfg);
  live_analysis(cfg);
  optimize(cfg);
}
```

Extracting Distant-ILP for Emerging Machines

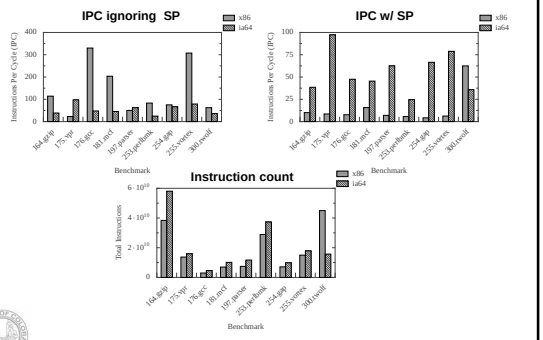


- *All chips will have multiple processor cores*
  - Chip multiprocessor (CMP) and chip multi-threaded (CMT)
  - Potentially multithreaded (SMT/MT) in each core
    - Lower communication costs
    - Purpose-built low-latency communication hardware
    - Inexpensive speculation (e.g, transactional memory)
- Challenges
  - Developing effective parallel programming models
  - Extracting parallelism from single-thread application for these machines

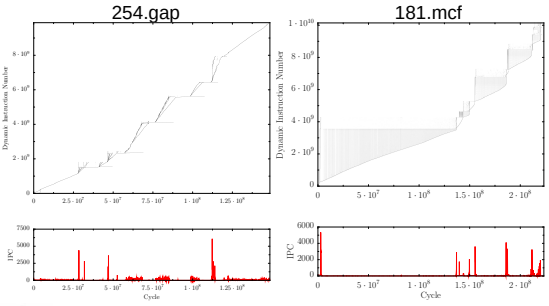
Methodology

- Gather execution traces for SPEC-CPU2000-INT
  - Optimized binaries (-O3) compiled using gcc and icc
  - Pin tool [Luk '05] dynamically instruments benchmark binaries to collect trace information
  - Traces are compressed with TCgen [Burtscher '04]
  - Architectures: Itanium, x86, PowerPC
- Schedule traces on an "ideal" machine using Adamantium
  - Only models true data dependences
  - Eliminates crippling dependences such as the stack pointer
  - Performs perfect branch prediction
  - Executes all instructions in a single cycle
  - Instruction window size can be varied

IPC Overview



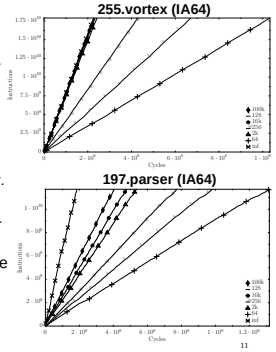
Amount of Distant ILP



Top Graph: Vertical lines indicate distant parallelism

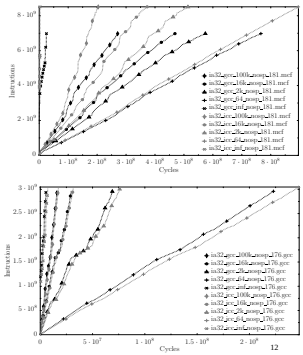
Effect of Instruction Window Size

- 255.vortex (top) maxes out at a 2k instruction window.
  - 300.twolf also had this behavior.
- 197.parser (bottom) exhibits unbounded increases in parallelism.
  - The remaining benchmarks we studied showed similar behavior.
- We found that ILP limitations for larger windows in benchmarks such as 255.vortex on IA64 were due to predicate dependences.



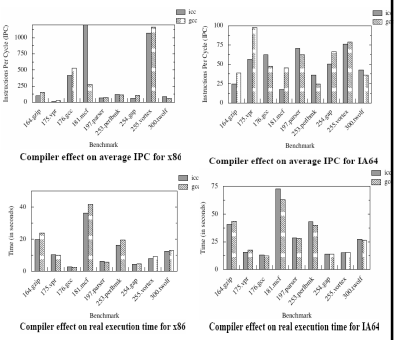
Effect of the Compiler on ILP

- More aggressive compiler expected to perform better.
- Compilers are designed to exploit nearby ILP.
- Better compiler might hurt its own chances of exploiting distant parallelism.



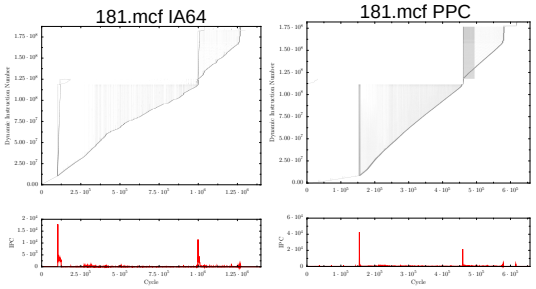
Compiler effect comparison

- For x86 - icc has a lower average IPC but is faster by executing fewer instructions.
- For Itanium - neither compiler seems to have an advantage.



13

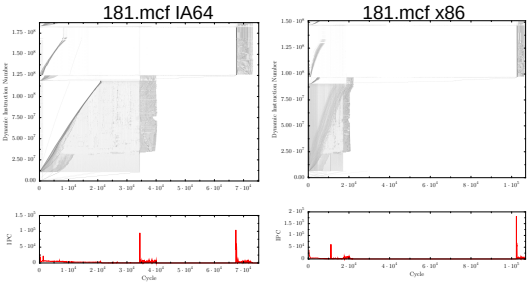
Comparing Architectures (IA64 vs. PPC)



- PPC with stack-pointer dependence versus Itanium with predicate register dependences

14

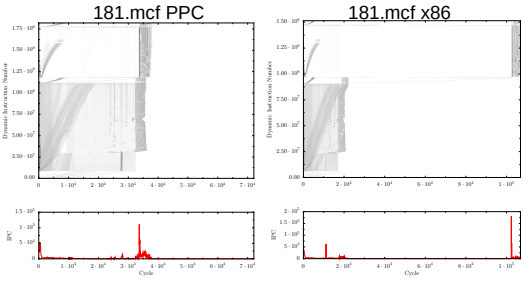
Comparing Architectures (IA64 vs. x86)



- IA64 without predicate register dependences versus x86 ignoring stack-pointer manipulations

15

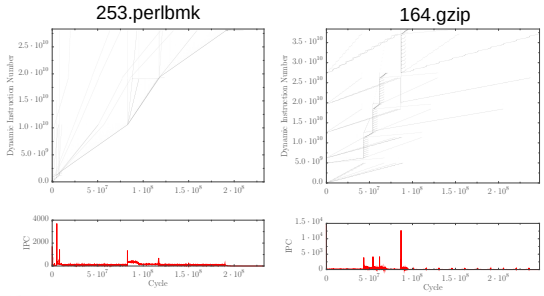
Comparing Architectures (PPC vs. x86)



- Chain of dependences at end of PPC
- Similar dependence chain in middle of x86

16

Independent Distant ILP on x86



17

Conclusions

- ILP differs based on ISA
  - When stack-pointer (x86 and PPC) and predicate dependences (IA64) are ignored, the overall distant ILP pattern is similar
- Distant ILP manifests in application specific ways
  - Critical dependences (dependences that uncover large amounts of parallelism)
  - Independent chains of localized dependences
- Compiler helps and hinders
  - Techniques to improve nearby ILP can harm distant ILP
- Future work
  - Critical dependences - can value prediction be used?
  - Independent chains - can separate threads be spawned?
  - Add realistic branch prediction and memory latency models
    - How much distant ILP can be harnessed in emerging CMP systems?
  - Resolve remaining differences among ISAs

18